Radon Series Comp. Appl. Math

Differential Elimination and Biological Modelling

François Boulier

Key words. Differential algebra. Elimination theory. Applications. Biology

AMS classification. 12H05, 37N25, 62P10

1 Introduction

This paper describes applications of a *computer algebra* method, *differential elimination*, to applied mathematics problems mostly borrowed from biology. The two con-

The author would like to thank Marc Lefranc for his comments and his advices.

Proceedings of the Workshop D2.2 of the *Special Semester on Groebner Bases and Related Methods*, May 8 - May 17, 2006, to appear in: *Groebner Bases in Symbolic Analysis*, RICAM Book Series, de Gruyter

sidered applications are related to the *parameters estimation* (chapter 3) and the *model reduction* (chapter 4) problems. In both cases, differential elimination can be viewed as a preparation to numerical treatments. Those numerical treatments are, at least partly, sketched in this paper in order to put some light on the real limitations of the applications. Together with the applications, the paper introduces two *implementations* of the differential elimination algorithms: the *diffalg* package, which is embedded in the MAPLE computer algebra software and the *BLAD* libraries [4] which are standalone open source C libraries. The *diffalg* package is designed to be manipulated interactively and can be used very quickly and easily by casual readers. The *BLAD* libraries are designed to provide differential elimination for scientific software independent of any computer algebra system. They are probably better suited than *diffalg* to the development of software dedicated to the described applications. Using the *BLAD* libraries implies however to write a C program. For this reason, in this paper, examples are illustrated with *diffalg* rather than with *BLAD*.

2 Differential Elimination

The three next sections can be read in any order and provide three different introductions to differential elimination: section 2.1 provides historical notes, section 2.2 presents it more algebraically, through the differential ideal membership problem while section 2.3 introduces it through software. For a wider survey on differential equations and computer algebra, see [68].

2.1 Historical Introduction

Differential elimination is an algorithmic subtheory of *differential algebra* (see section 2.2 for mathematical definitions). It solves the membership problem for radical differential ideals[1].

The membership problem for polynomial ideals was one of the main problems of commutative algebra. It was solved by Bruno Buchberger in [16], thanks to the theory of Gröbner bases. Similarly, the membership problem for differential ideals is one of the main problems of differential algebra. It is proven undecidable in general [33]. It is still open for finitely generated differential ideals. It is only solved in the special case of radical differential ideals.

The development of differential elimination was undertaken by Ritt who developed the concept of *characteristic sets*. In his book, Ritt gave an algorithm to decompose the radical of any finitely generated differential ideal as an intersection of finitely many differential prime ideals presented by characteristic sets[2]. Ritt's algorithm relies on factorizations over towers of algebraic extensions of the base field of the polynomials

[1] In this paper, differential ideals always refer to differential polynomial ideals.

[2] The intersection may be redundant. Surprisingly, the inclusion problem of two differential prime ideals presented by characteristic sets is still open while the equality test is straightforward [43, Chapter IV, Problem 3].

and does not cover the case of partial differential polynomials. Abraham Seidenberg designed in [66] an elimination algorithm for systems of differential polynomials which only relies on addition, multiplication and the equality test with zero in the base field of the polynomials. However, Seidenberg's method is not convenient: it takes as input a differential polynomial, a differential system and decides if the polynomial belongs to the radical of the differential ideal generated by the system. It does not provide a description of this radical differential ideal. It also involves some useless operations (e.g. computation of *preparation polynomials*). To cover the case of partial differential systems, Seidenberg developed an analogue of the S-polynomials theory of the Gröbner bases theory. However, the proof of his [66, Theorem VI] seems to be incomplete. A few years later, Azriel Rosenfeld fixed and generalized Seidenberg's Theorem VI in [62, Lemma] but did not provide any algorithm. In his book, Kolchin generalized "Rosenfeld's lemma" and described a generalized method [43, Section IV.9]. However, Kolchin's method involves some non effective steps: his approach cannot be treated as an algorithm. Later, Wu Wen-Tsün described in [70] an algorithm to decompose a given system of differential polynomials as finitely many characteristic sets but the characteristic sets in the sense of Wu are weaker than those of Ritt and are not sufficient (without any extra process) to decide membership in the radical of the differential ideal generated by the system. Dongming Wang developed Wu's method in [75].

Giuseppa Carra-Ferro and François Ollivier developed the concept of *differential Gröbner bases* in [19, 57] but the bases they define do not need to be finite. Elizabeth Mansfield developed another concept of *differential Gröbner bases* in [50] but Mansfield's bases do not solve the membership problem in differential ideals. Greg Reid developed the concept of *reduced involutive forms* together with an algorithm in [59]. This concept applies more generally to systems of analytic differential equations. In this setting, no satisfactory analogue of the Rosenfeld's lemma is however available.

The author developed the so-called *Rosenfeld-Gröbner* algorithm in [7] from the papers of Seidenberg and Rosenfeld. He used Gröbner bases to convert Rosenfeld's lemma into an algorithm[3]. *Rosenfeld-Gröbner* gathers as input a differential system and a *ranking*. It represents the radical of the differential ideal generated by the input system as a finite intersection of radical differential ideals presented by characteristic sets (in the sense of Ritt). It solves the membership problem to radical differential ideals (ordinary or with partial derivatives). It only relies on addition, multiplication and the equality test with zero in the base field of the polynomials. The algorithm described in [7] was much improved, theoretically and practically, by a lemma[4] due to Daniel Lazard[5] [9, Lemma 2]. See [13] for a survey on Lazard's lemma. Some variants of *Rosenfeld-Gröbner* were published afterwards [48, 40, 14, 41].

[3]Gröbner bases are no more involved in current implementations of *Rosenfeld-Gröbner*. Instead, a variant [10, 12, *RegCharacteristic*] of *LexTriangular* [46, 52] is used.

[4]Lazard's lemma is a non differential lemma which implies, when combined to Rosenfeld's lemma, that the differential ideals presented by characteristic sets are necessarily radical.

[5]There was a gap in the proof of "Lazard's lemma" in [9] which was fixed for the first time by Sally Morrison in [54, 55].

2.2 Algebraic Introduction

Differential algebra is an algebraic theory for differential equations (ordinary or with partial derivatives) which was founded by Joseph Fels Ritt in the first half of the twentieth century. Ritt was much impressed by the development of commutative algebra and wanted to achieve a similar theory for differential equations. He summarized the work of his team in [61]. One of his students, Ellis Robert Kolchin, developed still further Ritt's theory and summarized his results and that of his team in [43]. See [18] for a survey. A *differential ring* (resp. field) is a ring (resp. field) R endowed with a derivation (this paper is restricted to the case of a single derivation but the theory is more general) i.e. a unitary mapping $R \to R$ such that (denoting $\dot{a}$ the derivative of a):

$$\dot{(a+b)} = \dot{a} + \dot{b}, \quad \dot{(a\,b)} = \dot{a}\,b + a\,\dot{b}.$$

Observe that, theoretically, the derivation is an abstract operation. For legibility, one views it as the derivation w.r.t. the time t. Algorithmically, one is led to manipulate finite subsets of some *differential polynomial ring* $R = K\{U\}$ where K is the differential field of coefficients (in practice, $K = \mathbb{Q}$ or $K = \mathbb{Q}(t)$) and U is a finite set of *dependent variables*[6]. The elements of R, the *differential polynomials* are just polynomials in the usual sense, built over the infinite set, denoted ΘU, of all the derivatives of the dependent variables.

A famous example of Ritt [61, Section II.4]. The left-hand side of the ordinary differential equation $\dot{u}^2 - 4\,u = 0$ is a differential polynomial of the differential polynomial ring $R = \mathbb{Q}\{u\}$. Its analytic solutions are the zero function $u(t) = 0$ and the family of parabolas $u(t) = (t+c)^2$ where c is an arbitrary constant.

Definition 2.1 A *differential ideal* of a differential ring R is an ideal of R, stable under the action of the derivation.

The study of the radical of the differential ideal generated[7] by a finite system of differential polynomials is strongly related to the study of the analytic solutions of this system. Indeed, in algebraic geometry, it is well known that the set of the polynomials which vanish over the solutions of a given polynomial system form an ideal and even a radical ideal [78, Section VII.3, Theorem 14]. For differential equations, the set of the differential polynomials which vanish over the analytic[8] solutions of a given differential polynomial system form a differential ideal and even a radical differential ideal [61, Sections II.4 and II.7].

[6] In the differential algebra theory, the terminology *differential indeterminates* is preferred to *dependent variables* for derivations are abstract and differential indeterminates are not even assumed to correspond to functions. In order not to mix different expressions in this paper, the second expression, which seems to be more widely known, was chosen.

[7] An ideal $\mathfrak{A}$ is said to be *radical* if $a \in \mathfrak{A}$ whenever there exists some nonnegative integer p such that $a^p \in \mathfrak{A}$. The radical of an ideal $\mathfrak{A}$ is the set of all the ring elements a power of which belongs to $\mathfrak{A}$. The radical of a (differential) ideal is a radical (differential) ideal [65, Section 4].

[8] Over some unspecified domain.

Ritt's example (continued). The analytic solutions of the differential equation $\dot{u}^2 - 4\,u = 0$ are the function $u(t) = 0$ and the family of functions $u(t) = (t + c)^2$. These solutions are also solutions of all the derivatives of the differential equation:

$$2\,\dot{u}\,(\ddot{u} - 2) = 0, \quad 2\,\dot{u}\,\dddot{u} + 2\,\ddot{u}\,(\ddot{u} - 2) = 0, \quad \ldots$$

More generally, they are solutions of every differential polynomial, a power of which is a finite linear combination of the derivatives of $\dot{u}^2 - 4\,u$ with arbitrary differential polynomials as coefficients i.e. every element of the radical of the differential ideal generated by $\dot{u}^2 - 4\,u$.

The problem of computing a representation of the radical of the differential ideal generated by a finite set of differential polynomials is thus an important problem, related to the study of the analytic solutions of this system. So is the membership problem to radical differential ideals which is solved by *Rosenfeld-Gröbner*. To present it, one needs to define the concept of *ranking* and Ritt's reduction.

Definition 2.2 If U is a finite set of dependent variables, a *ranking* over U is a total ordering over the set ΘU of all the derivatives of the elements of U which satisfies: $a < \dot{a}$ and $a < b \Rightarrow \dot{a} < \dot{b}$ for all $a,\, b \in \Theta U$.

Let U be a finite set of dependent variables. A ranking such that, for every $u,\, v \in U$, the ith derivative of u is greater than the jth derivative of v whenever $i > j$ is said to be *orderly* [43, Section I.8]. If U and V are two finite sets of differential variables, one denotes $U \gg V$ every ranking such that any derivative of any element of U is greater than any derivative of any element of V. Such rankings are said to *eliminate* U w.r.t. V.

Definition 2.3 Assume that some ranking is fixed. Then one may associate with any differential polynomial $f \in K\{U\} \setminus K$ the greatest (w.r.t. the given ranking) derivative $v \in \Theta U$ such that $\deg(f,\, v) > 0$. This derivative is called the *leading derivative* or the *leader* of f.

Ritt's reduction. It is a generalization of the Euclidean division. It is well known that, if f and g are two polynomials, in one variable v, with coefficients in a field, the Euclidean division of f by g (g nonzero) is possible. It yields a unique pair $(q,\, r)$ of polynomials such that $f = g\,q + r$ and $\deg r < \deg g$. If f and g have coefficients in a ring, the Euclidean division is no more possible in general for the leading coefficient of g may not be invertible. The closest available algorithm is the *pseudodivision* which consists in multiplying f by the leading coefficient c of g, raised at the power $p = \deg f - \deg g + 1$ before performing the Euclidean division [73, Section 6.12]. It yields a unique pair $(q,\, r)$ of polynomials such that $c^p\, f = g\,q + r$ and $\deg r < \deg g$. The polynomial r is called the *pseudoremainder* of f by g and is denoted $\mathrm{prem}(f,\, g)$ or $\mathrm{prem}(f,\, g,\, v)$ when the variable is not clear from the context (case of polynomials depending on many different variables). The pseudodivision generalizes to the differential setting, providing Ritt's reduction algorithm [43, Section I.9], described below. Observe that only the "remainder" is computed.

Let f be a differential polynomial, to be reduced by a finite set $C = \{g_1, \ldots, g_n\}$ of differential polynomials. Denote v_i the leader of g_i for $1 \leq i \leq n$ (assuming that none of the g_i lies in the base field). Ritt's reduction builds a sequence $f_0, \ldots, f_r$ of differential polynomials starting at $f_0 = f$. The result is the polynomial

$$f_r = \mathrm{Ritt\_reduction}(f, C).$$

To compute $f_{\ell+1}$ from f_ℓ, three cases may occur. First case: if, for each $1 \leq i \leq n$, the differential polynomial f_ℓ does not depend on any proper derivative[9] $v_i^{(k)}$ of v_i and $\deg(f_\ell, v_i) < \deg(g_i, v_i)$ then the computation stops and $f_\ell = f_r$ is returned. Second case: if there exists some index $1 \leq i \leq n$ such that $\deg(f_\ell, v_i) \geq \deg(g_i, v_i)$ then $f_{\ell+1} = \mathrm{prem}(f_\ell, g_i, v_i)$. Third case: if there exists some index $1 \leq i \leq n$ such that f_ℓ depends on some proper derivative $v_i^{(k)}$ of v_i then $f_{\ell+1} = \mathrm{prem}(f_\ell, g_i^{(k)}, v_i^{(k)})$.

Remarks. The second rule could actually be viewed as a particular case of the third one. The sequence $f_0, \ldots, f_r$ described above is not uniquely defined. One could define a precise algorithm by specifying that the sequence of the reduced derivatives $v_i^{(k)}$ must be decreasing. This is the usual strategy but any other strategy could be applied. Last, observe that whenever $k \geq 1$, the differential polynomial $g_i^{(k)}$ has degree one in $v_i^{(k)}$ and admits the *separant* $s_i = \partial g_i / \partial v_i$ for leading coefficient. In this case, writing $g_i^{(k)} = s_i\, v_i^{(k)} + t_{i,k}$, one sees that the pseudodivision of f_ℓ by $g_i^{(k)}$ amounts to the following: first perform the following substitution in f_ℓ

$$v_i^{(k)} \longrightarrow -\frac{t_{i,k}}{s_i}$$

then clear the denominator of the obtained rational fraction. The resulting polynomial is free of $v_i^{(k)}$.

Example. Let us apply Ritt's reduction over $f_0 = \ddot{u} - v\,\dot{u}$ and $C = \{\dot{u}^2 + v\}$. The ranking is $u \gg v$ so that the leader of $g = \dot{u}^2 + v$ is $\dot{u}$. The polynomial f_0 gets pseudoreduced by the first derivative of g i.e. $2\,\dot{u}\,\ddot{u} + \dot{v}$. First one substitutes $\ddot{u} \longrightarrow -\dot{v}/(2\,\dot{u})$ over f_0, giving the rational fraction

$$-\frac{\dot{v}}{2\,\dot{u}} - v\,\dot{u}.$$

Second, the denominator is cleared, giving $f_1 = -\dot{v} - 2\,v\,\dot{u}^2$. This polynomial f_1 gets pseudoreduced by g: one substitutes $\dot{u}^2 \longrightarrow -v$ over f_1, giving the differential polynomial f_2 (there is no denominator to clear).

$$f_2 = -\dot{v} + 2\,v^2.$$

Ritt's reduction stops at this step and $f_2 = f_r$ is returned.

[9] One denotes $v_i^{(k)}$ the kth derivative of v. When $k \geq 1$, $v_i^{(k)}$ is said to be a *proper* derivative of v_i. When $k = 0$, one defines $v_i^{(k)} = v_i$.

Normal forms. Observe that in general, the set of all the differential polynomials which are reduced to zero by Ritt's reduction has no clear structure. It does not even need to be an ideal. Observe also that the returned polynomial f_r is not equivalent to f modulo the differential ideal generated by C because of the denominator clearing step. A more careful version was designed in [12]. It returns a rational fraction instead of a polynomial. When C is a *characteristic set* of the ideal $\mathfrak{A}$ that it defines, the rational fraction is guaranteed to be a *normal form* of the residue class of f modulo $\mathfrak{A}$. Such a normal form algorithm may be used to detect linear dependencies between residue classes modulo $\mathfrak{A}$, following the idea of [29]. See [8] or [5, Section 6.1].

Rosenfeld-Gröbner. The *Rosenfeld-Gröbner* algorithm gathers as input a finite system F of differential polynomials and a ranking. It returns a finite family (possibly empty) $C_1, \ldots, C_r$ of finite subsets of $K\{U\} \setminus K$. Each system C_i defines a differential ideal $\mathfrak{C}_i$ in the sense that, for any $f \in K\{U\}$, we have

$$f \in \mathfrak{C}_i \quad \text{iff} \quad \mathrm{Ritt\_reduction}(f, C_i) = 0.$$

The relationship with the radical $\mathfrak{A}$ of the differential ideal generated by F is the following:

$$\mathfrak{A} = \mathfrak{C}_1 \cap \cdots \cap \mathfrak{C}_r.$$

When $r = 0$ we have $\mathfrak{A} = K\{U\}$. Combining both relations, one gets an algorithm to decide membership in $\mathfrak{A}$. Indeed, given any $f \in K\{U\}$ we have:

$$f \in \mathfrak{A} \quad \text{iff} \quad \mathrm{Ritt\_reduction}(f, C_i) = 0, \quad 1 \leq i \leq r.$$

The systems C_i are often called *(differential) characteristic sets* or *differential regular chains*[10] in the literature. The differential ideals $\mathfrak{C}_i$ do not need to be prime. They are however necessarily radical, thanks to Lazard's lemma. Observe that it is possible to refine further the intersection in order to get prime differential ideals. It is sufficient for this to apply a usual primary decomposition algorithm. However, no algorithm is known to decide inclusion between differential ideals presented by characteristic sets, even when they are prime [43, Section IV.9, Problem 3]. Thus the computed representation can by no means be guaranteed to be minimal though this latter theoretically exists.

Ritt's example (continued). When $U = \{u\}$ there exists only one ranking:

$$\cdots > \ddot{u} > \dot{u} > u.$$

Take $F = \{\dot{u}^2 - 4\,u\}$ and denote $\mathfrak{A}$ the radical differential ideal generated by F. If one applies the *Rosenfeld-Gröbner* to F and this ranking, one gets an intersection $\mathfrak{A} = \mathfrak{C}_1 \cap \mathfrak{C}_2$ with

$$C_1 = \{\dot{u}^2 - 4\,u\}, \qquad C_2 = \{u\}.$$

[10] There is a slight difference between these two notions but it does not matter in this paper.

The differential polynomial u is reduced to zero by C_2, not by C_1. Thus $u \notin \mathfrak{A}$. The differential polynomial $\ddot{u} - 2$ is reduced to zero by C_1, not by[11] C_2. Thus $\ddot{u} - 2 \notin \mathfrak{A}$. The product $\dot{u}\,(\ddot{u}-2)$ is reduced to zero by C_1 and C_2. Thus it lies in $\mathfrak{A}$ (it is one-half of the first derivative of $\dot{u}^2 - 4\,u$). This proves that the ideal $\mathfrak{A}$ is not prime. The ideal $\mathfrak{C}_1$ corresponds to the family of parabolas $u(t) = (t+c)^2$. The ideal $\mathfrak{C}_2$ corresponds to the solution $u(t) = 0$.

Complexity. From a theoretical point of view, differential elimination is a very powerful tool. It permits to decide if a system of differential equations admits analytic solutions over some unspecified domain[12]. See [67, Embedding theorem] and [60, 47]. Moreover, non differential polynomial elimination can be reduced to differential elimination in two different ways. First any non differential polynomial system can be viewed as a differential system of order zero (one seeks constant functions solutions instead of numbers) and the differential characteristic sets computed by *Rosenfeld-Gröbner* are exactly those that non differential algorithms [45, 42, 53] would compute. Second, any non differential polynomial system can be encoded as a system of linear partial differential equations in one dependent variable and constant coefficients ; the differential characteristic set computed by *Rosenfeld-Gröbner* over this linear system is (up to the inverse encoding) the reduced Gröbner basis of the non differential system w.r.t. the admissible ordering induced by the ranking. This last reduction proves that the membership problem to radical differential ideal is exspace hard [44]. See also [5, Section 9.7].

2.3 Computational Introduction

There are many different ways to tackle systems of ordinary differential equations in a computer algebra software. *Differential elimination* is one of them. It is presented here by comparison with numerical integration and closed form integration and illustrated over the *differential index reduction* problem. Most computations are performed using the *diffalg* package of MAPLE 9. A short presentation of the *BLAD* libraries is provided too.

Numerical integration. Here is an example of an ordinary differential equation with an initial condition. The dependent variable x represents an unknown time varying function (one denotes $\dot{x}$ the first derivative of x).

$$\dot{x} = x\,(3 - x), \qquad x(0) = 1.$$

Numerical integration of an ordinary differential equation with an initial condition consists in computing a discrete approximation of the graph of the integral curve of the equation as a finite number of points. In principle, it is always possible to carry it out. The simplest method is Euler's explicit method [36, page 132]. Numerical integration

[11] Proving that $\mathfrak{C}_1 \not\subset \mathfrak{C}_2$ though C_1 is reduced to zero by C_2.

[12] One encounters undecidability results when the domain is precised. See [21, Theorem 4.11].

is not considered as a method of computer algebra. The commands below show how to numerically integrate the above example using MAPLE 9 (the method is not the one of Euler but an adaptative stepsize Runge-Kutta scheme). The output of the numerical integrator is a function which evaluates the solution.

```
ode := diff(x(t),t) = x(t)*(3-x(t));
```

$$ode := \frac{d}{dt}x(t) = x(t)(3 - x(t))$$

```
sol := dsolve ({ode, x(0)=1}, x(t), numeric):
sol (0.5);
```

$$[t = 0.5,\ x(t) = 2.07431460567341386]$$

Closed form integration. *Closed form integration* of an ordinary differential equation consists in computing its solutions as finite formulae. See [15] for an introductory text. Over the example, it is possible and yields the formula below. Observe that the formula involves an arbitrary constant *_C1* for no initial condition is specified. Closed form integration is part of computer algebra. It is however not possible in general. It is different from differential elimination.

```
dsolve (ode, x(t));
```

$$x(t) = \frac{3}{(1 + 3\,e^{-3\,t}\_C1)}$$

Differential elimination. To explain what *differential elimination* is, one needs to consider a system of at least two ordinary differential equations. The following example is borrowed from [37, Chapter VII, page 454]. Since it mixes ordinary differential equations and non differential equations, this type of system is sometimes called a *differential algebraic system*[13]. There are three unknown time varying functions (three dependent variables) x, y and z :

$$\begin{aligned} \dot{x} &= 0.7\,y + \sin(2.5\,z), \\ \dot{y} &= 1.4\,x + \cos(2.5\,z), \\ 1 &= x^2 + y^2. \end{aligned}$$

Even readers not familiar with differential algebraic systems may see that such systems raise problems. Assume that some initial conditions $x(0)$, $y(0)$ and $z(0)$ are given and let us try to numerically integrate the system with Euler's method for some stepsize h. Evaluating the right-hand sides of the two first equations at $t = 0$ one gets $\dot{x}(0)$ and $\dot{y}(0)$. Using these numbers, Euler's method permits us to compute the estimations $x(h) \simeq x(0) + h\,\dot{x}(0)$ and $y(h) \simeq y(0) + h\,\dot{y}(0)$. However, one cannot estimate

[13] For readers familiar with this notion, it has *differentiation index* 2 [37, Section VII.1, Definition 1.2].

the value of $z(h)$ since no ordinary differential equation of the form (2.1) is available. Thus Euler's method cannot perform the next step.

$$\dot{z} \quad = \quad \textit{something} \tag{2.1}$$

The point here is that the ordinary differential equation (2.1) which seems to be missing is actually not missing but hidden in some differential ideal[14]. It can be automatically extracted from the initial system by means of *differential elimination*. Before showing how to proceed with the help of the *diffalg* package of MAPLE, one needs to convert the system as a *polynomial differential system*. For this, one denotes s the sine, c the cosine and one introduces a few more equations. The following differential polynomial system is equivalent to the above one.

$$\begin{array}{rclcrcl} \dot{x} & = & 0.7\,y + s, & & \dot{s} & = & 2.5\,\dot{z}\,c, \\ \dot{y} & = & 1.4\,x + c, & & \dot{c} & = & -2.5\,\dot{z}\,s, \\ 1 & = & x^2 + y^2. & & 1 & = & s^2 + c^2. \end{array}$$

Let's now compute the hidden equation using *diffalg*. One first stores the differential polynomial system in the variable *syst*, converting floating point numbers as rational numbers.

```
with (diffalg):
syst := [diff(x(t),t) - 7/10*y(t) - s(t),
         diff(y(t),t) - 14/10*x(t) - c(t),
         x(t)^2 + y(t)^2 - 1,
         diff(s(t),t) - 25/10*diff(z(t),t)*c(t),
         diff(c(t),t) + 25/10*diff(z(t),t)*s(t),
         s(t)^2 + c(t)^2 - 1]:
```

Then one assigns to the variable R the context of the computation: one indicates that the only derivation is taken with respect to the time, that the notation is the standard *diff* notation of MAPLE and one provides the *ranking*. For short[15], let us just say that the fact that z stands on the rightmost place of the list indicates that we are looking for an ordinary differential equation of the form (2.1).

```
R := differential_ring (derivations = [t], notation = diff,
                        ranking = [[s, c, x, y, z]]):
```

Next the *Rosenfeld-Gröbner* function is applied to *syst* and R. It returns a list of MAPLE tables. Each table provides a *characteristic set*. The list should be understood as an intersection. Over the example, the list only involves one characteristic set so that the characteristic set does represent the radical differential ideal generated by the input system. The desired equation stands on the second place of the characteristic set (only the two first equations are displayed). Enlarging the input system with this equation, it is now easy to perform any numerical integration method and our problem is solved. Technically speaking, differential elimination has permitted the reduction to zero of the *differentiation index* of the input system: it was 2 ; it is now 0. See [31, 58] for related works.

[14] All the differential algebra terminology used in this section is precisely defined in section 2.2.

[15] With the terminology inroduced in section 2.2, this is the *orderly* ranking such that $s > c > x > y > z$.

```
ideal := Rosenfeld_Groebner (syst, R):
rewrite_rules (ideal [1]);
```

$$\left[\frac{d}{dt} y(t) = \frac{7}{5} x(t) + c(t), \right.$$

$$\frac{d}{dt} z(t) = \frac{1}{25} \frac{3500 - 12348\,(y(t))^6 + 13230\, c(t)\, x(t)\,(y(t))^4 + 25809\,(y(t))^4}{441\,(y(t))^6 - 882\,(y(t))^4 + 541\,(y(t))^2 - 100}$$

$$\left. + \frac{1}{25} \frac{-14700\, x(t)\,(y(t))^2\, c(t) - 16961\,(y(t))^2 + 3940\, x(t)\, c(t)}{441\,(y(t))^6 - 882\,(y(t))^4 + 541\,(y(t))^2 - 100}, \quad \cdots \right]$$

Let us now perform some slight change on the chosen ranking. Strictly speaking, the ranking below is different from the above one[16] but it also indicates that we are looking for an ordinary differential equation of the form (2.1). However, if one applies *Rosenfeld-Gröbner* over *syst* for this ranking, one never gets any result because of the size of the equations the algorithm tries to compute.

```
R := differential_ring (derivations = [t], notation = diff,
                        ranking = [[s, c, x, y], z]):
ideal := Rosenfeld_Groebner (syst, R):
```

Warning, computation interrupted

To summarize, differential elimination is a process which takes as input a system of differential equations (ordinary or with partial derivatives) and a ranking. It rewrites the input system into another equivalent system (or an equivalent finite family of systems when case splittings are necessary). The ranking permits to control the elimination process, indicating what should be eliminated. Differential elimination methods are considered as computer algebra. In principle, differential elimination is always possible. However, in practice, it is restricted by its terrifying worst case complexity and the related problem of choosing rankings.

A few packages are available for differential elimination: the *diffgrob* package of Mansfield [50], the *rif* package of Reid, Wittkopf and Boulton [59], the *epsilon* package of Wang [76] and the *diffalg* package which was illustrated just above. The first version of the *diffalg* package was written by the author in 1995 for MAPLE 5 [7, 9]. However, the version involved in MAPLE 9 is not the original one since it was much improved by Évelyne Hubert [40] and, more recently, by François Lemaire [12].

[16] It is the ranking $(s,\, c,\, x,\, y) \gg z$ which eliminates s, c, x and y and such that $s > c > x > y$.

The BLAD libraries. In order to overcome (at least partially) the difficulties stated above, the author has developed a C library, called *BLAD*, from the model of the *GMP* library. This library aims at providing differential elimination methods to scientific software which are not necessarily computer algebra systems. It is available on [4]. One of the important functionnalities it provides consists in bounding in advance the time and the memory allocated to a given differential elimination request. In the case of a failure, the calling program gets back a clean working environment. The following C program performs the first elimination provided above. It reads the data in characters strings and prints the result of the differential elimination on the standard output. Of course, this is not a natural way to use the *BLAD* libraries.

```
#include "bad.h"

int main ()
{   struct bad_intersectof_regchain ideal;
    struct bap_tableof_polynom_mpz eqns, ineqns;
    bav_Iordering r;

    bad_restart (0, 0);
    ba0_sscanf2
      ("ordering (derivations = [t], blocks = [[s, y, c, x, z]])",
       "%ordering", &r);
    bav_R_push_ordering (r);
    bad_init_intersectof_regchain (&ideal);
    ba0_sscanf2
       ("intersectof_regchain ([], \
            [differential, primitive, autoreduced, normalized])",
        "%intersectof_regchain", &ideal);
    ba0_init_table ((ba0_table)&eqns);
    ba0_init_table ((ba0_table)&ineqns);
    ba0_sscanf2 ("[10*x[t] - 7*y - 10*s, 10*y[t] - 14*x - 10*c, \
                   10*s[t] - 25*z[t]*c, 10*c[t] + 25*z[t]*s, \
                   x^2 + y^2 - 1, c^2 + s^2 - 1]",
                 "%t[%Az]", &eqns);
    bad_Rosenfeld_Groebner (&ideal, &eqns, &ineqns, 0);
    ba0_printf ("%intersectof_regchain\n", &ideal);
    bad_terminate (ba0_init_level);
    return (0);
}
```

There are four stacked *BLAD* libraries. From top down: *bad* (differential elimination), *bap* (differential polynomials), *bav* (rankings) and *ba0* (kernel). Functions identifiers are prefixed by the library they belong to. The *main* function starts by defining some variables: *ideal* which is going to contain the result, *eqns* and *ineqns* which will serve to store the input system and r which will contain the ranking. The first instruction (*bad_restart*) starts a *sequence of calls* to the library. This sequence terminates with the call to *bad_terminate*. The two parameters provided to *bad_restart* give the limits, in time and in memory, allocated to the sequence of calls. A zero parameter means that there is no limit. Then the ranking is read from a string and stored in r (the *ba0_sscanf2*

function provides a generalization of the *sscanf* function of the standard C library). The variable *ideal* is initialized to an empty intersection of regular differential chains (characteristic sets) endowed with some attributes which will serve to parametrize the elimination: *"differential"* indicates that the ideal represented by the variable is differential, the other attributes set some technical properties that the regular differential chains will have to satisfy. Then the array *eqns* is initialized with the system to process (*x[t]* denotes $\dot{x}$). We do not need to bother with *ineqns* which is not used here. Last *Rosenfeld-Gröbner* is called and the content of *ideal* is printed on the screen. Here is the result of the execution. The desired equation starts on the third line.

```
intersectof_regchain ([regchain ([100*c^2 - 420*c*x^3 + 420*c*x -
441*x^4 + 341*x^2, y^2 + x^2 - 1, 10*s*x + 10*y*c + 21*y*x,
11025*z[t]*x^5 - 11025*z[t]*x^3 + 2500*z[t]*x + 13230*c*x^4 -
11760*c*x^2 + 2470*c + 12348*x^5 - 11235*x^3 + 2387*x, 5*x[t]*x
+ 5*y*c + 7*y*x], [differential, autoreduced, primitive,
squarefree, coherent, normalized])], [differential, autoreduced,
primitive, squarefree, coherent, normalized])
```

3 Parameters Estimation

This section describes an application of differential elimination and, more precisely, an application of algorithms which perform changes of rankings over characteristic sets. The principle of this application was designed by Ghislaine Joly-Blanchard, Lilianne Denis-Vidal and Céline Noiret [23] and presented in [56]. The addressed problem is this one: estimate parameters values of parametric ordinary differential systems the dependent variables of which are not all *observed*. When all the dependent variables of the system are observed, the method still works but differential elimination is no more necessary. The work of Joly-Blanchard, Denis-Vidal and Noiret is strongly related to the problem of the *identifiability* study of differential systems, for which a huge literature is available. See e.g. [74, 30, 57, 24, 26, 25, 49, 2, 64]. The method of Joly-Blanchard, Denis-Vidal and Noiret is original for two reasons: it relies on rigorous differential elimination methods and it carries out the study of real examples up to the final numerical treatment. It mixes symbolics and numerics.

It assumes that the phenomenon under study is quite accurately modelled and that quite precise measures are available for the observed variables. Thus, though it was applied with quite some success in pharmacokinetics [20, 71], biological modelling may not be the most suitable field of application of the method. The method is described over an example coming from biology anyway, but it is more presented as an academic challenge than as a real application.

Here is a summary of the rest of this section. The addressed problem is stated over an example. The classical numerical solution is recalled. it relies on the use of a numerical nonlinear least squares solver i.e. a Newton method. Differential elimination gets involved in the process to help solving the most difficult part of the Newton method: guessing the starting point. Last the difficulties of the overall method are discussed.

3.1 Statement of the Problem over an Example

Figure 3.1 represents a *compartmental model*. The two *compartments* represent the blood and some organ. A medical product is injected in the blood at $t = 0$. It can go from the blood to the organ and conversely. It may also get degraded and exit from the system. In order to write the corresponding differential system, some hypotheses must be made on the nature of the exchanges: exchanges between the two compartments are assumed to be linear i.e. that, over every small enough interval of time, the amount of product going from compartment i to compartment j is proportional to the concentration of product in compartment i. The proportionality constant is denoted k_{ij}. The degradation is assumed to follow a *Michaelis-Menten* law. This law is a bit more difficult to explain. It can be derived from the modelling of an enzyme-catalyzed reaction by means of some *model reduction*. Two parameters are associated to this degradation: a maximal speed V_e and another constant k_e.

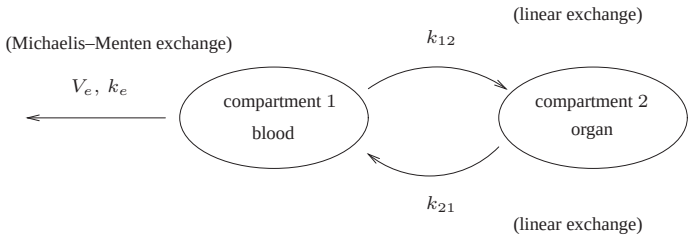

Figure 3.1 Compartmental model

From Figure 3.1, it is possible to derive a system of parametric ordinary differential equations. One associates to compartments 1 and 2, dependent variables x_1 and x_2 which represent the concentrations of product present in these compartments. Differential equations are built by considering exchanges the ones after the other ones. Each exchange appends one term to the right-hand side of the differential equation of the source compartment (with a minus sign) and a term to the right-hand side of the differential equation of the target compartment (with a plus sign). Beware to the trap: quantities are conserved by exchanges while exchanges are defined from the concentrations, which depend on the volumes of the compartments. For simplicity, it is assumed here that both compartments have a unitary volume. Applying the above process, one gets the following differential system. The second one is either linear or polynomial (it depends the way parameters are viewed). The first one is a rational fraction but it is equivalent to a polynomial since its denominator cannot vanish: parameters and dependent variables are positive real numbers.

$$\begin{aligned} \dot{x}_1 &= -k_{12}\,x_1 + k_{21}\,x_2 - \frac{V_e\,x_1}{k_e + x_1}, \\ \dot{x}_2 &= k_{12}\,x_1 - k_{21}\,x_2. \end{aligned}$$

Let us consider now some instance of the above model and assume that some extra information is available: parameters k_{12} and k_{21} are completely unknown, an interval of

possible values $70 \leq V_e \leq 110$ is known for V_e and that $k_e = 7$ is known[1] Some information is available on compartments also: compartment 1 is assumed to be *observed* i.e. a file of measures is assumed to be available for x_1. Compartment 2 is assumed to be non observed. One just knows that[2] $x_2(0) = 0$ i.e. that no product is initially present in the organ. To fix ideas and help the reader to reproduce the example studied in this section, here is a part of a file of 31 measures[3] for x_1.

```
     t          x1
0.00000e-01 5.00000e+01
5.00000e-02 4.45078e+01
    ...          ...
1.50000e+00 4.95270e-02
```

We are now ready to state the problem over this example: *given the system of parametric ordinary differential equations, the file of measures and the extra information, estimate the values of the three unknown parameters:* V_e, k_{12} *and* k_{21}.

3.2 The Numerical Method

There exists a purely numerical method to solve this problem. It is a *non linear* least squares solving method i.e. a Newton method. Precisely, a Levenberg-Marquardt solver is called. The idea is simple: pick *random* values for the three unknown parameters. Integrate numerically the differential system w.r.t. these values and compare the curve obtained by simulation with the file of measures. The *error* is defined as the sum, for all abscissas, of the squares of the ordinates differences between the two curves. The Levenberg-Marquardt method updates the values of the three unknown parameters if the error is considered as too large. It stops either if the error is small enough of if a stationary point is reached.

Let us try and take the following values: $V_e = 70$, $k_{12} = 4.5$ and $k_{21} = 1.5$. One gets the two curves on the left-hand side picture of Figure 3.2. After a few loops, the Levenberg-Marquardt yields the two curves on the right-hand side picture with $V_e = 82.8$, $k_{12} = .76$ and $k_{21} = .16$. Numerical computations (numerical integration of ODE, Levenberg-Marquardt method) were performed by the *Gnu Scientific Library* (*GSL*). The picture was produced by *gnuplot*. According to the pictures, the purely numerical method seems to work perfectly. However, the obtained parameters values are wrong: the Levenberg-Marquardt ended in a local minimum.

[1] It is realistic to assume that one of the parameters is known since equations can often be normalized by dividing some of the system parameters by one of them or, more generally, by studying their Lie symmetries.

[2] Unknown initial conditions do not raise any problem for they can be handled as plain parameters. See e.g. [36, Section I.14].

[3] The file was produced by numerical integration with $x_1(0) = 50$, $x_2(0) = 0$, $V_e = 101$, $k_{12} = 0.5$ and $k_{21} = 3$. The time ranges from $t = 0$ to $t = 1.5$ by steps of length 0.05.

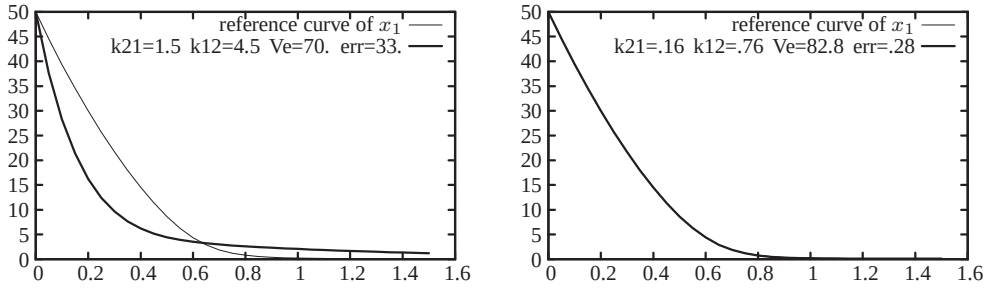

Figure 3.2: The reference (thin) and the simulated (thick) curves, before (left) and after (right) running the Levenberg-Marquardt method

3.3 The Symbolic-Numeric Method

The previous method has a drawback: it relies on *nonlinear* least squares which require the *a priori* knowledge of a good approximation of the parameters values. Thanks to differential elimination and to *linear* least squares, it is possible to estimate a first approximation of the parameters values. This first approximation may be used afterwards by the purely numerical method as a starting point.

The idea here consists in *eliminating the non observed variables* of the model. In other words, the idea consists in computing a differential polynomial which lies in the differential ideal generated by the model equations and which only involves the observed variable x_1, its derivatives up to any order and the model parameters. Let us show how to do this with the help of *diffalg*.

To compute this polynomial, the *Rosenfeld-Gröbner* algorithm is applied over the model equations. The ranking eliminates x_2 w.r.t. x_1:

$$x_2 \gg x_1.$$

In other words, the ranking indicates that we are looking for a polynomial free of x_2. The right-hand side of the first model equation is a rational fraction. It is decomposed as a numerator and a denominator. The numerator is stored in the list of the equations (first parameter to *Rosenfeld-Gröbner*). The denominator is stored in the list of the *inequations*[4] (second parameter to *Rosenfeld-Gröbner*). To avoid splitting cases on parameters values, one views them as (transcendental) elements of the base field of the differential polynomials.

```
K := field_extension
             (transcendental_elements = [k21, k12, ke, Ve]):
R := differential_ring
             (derivations = [t], notation = diff,
              field_of_constants = K, ranking = [x2, x1]):
```

[4]Inequations are polynomials which are considered as invertible. Indeed, if h is an inequation and some polynomial $h\,p$ lies in the ideal then p lies in the ideal. The ideal theoretic corresponding operation is the *saturation*.

```
ideal := Rosenfeld_Groebner
             ([numer (eq1), eq2], [denom (eq1)], R);
```

$$\textit{ideal} := [\textit{characterizable}]$$

The characteristic set *ideal* involves two polynomials. The one which does not involve x_2 is the second one, which is displayed below, slightly pretty printed. The expressions enclosed between square brackets are called *parameters blocks*.

$$\ddot{x}_1\,(x_1+k_e)^2+[k_{12}+k_{21}]\,\dot{x}_1\,(x_1+k_e)^2+[V_e]\,\dot{x}_1\,k_e+[k_{21}\,V_e]\,x_1\,(x_1+k_e)=0.$$

This equation tells us that the model is *globally identifiable* i.e. that, given a function x_1 and a parameter value k_e, the three unknown parameters are uniquely defined. Indeed, assume that the function x_1 is known. Then so are its derivatives $\dot{x}_1$ and $\ddot{x}_1$. These three functions can therefore be evaluated for three different values of the time t. The known parameter k_e can be replaced by its value. One thereby gets an exactly determined system of three linear equations whose unknowns are the parameters blocks. This system admits a unique solution. The values of the parameters blocks being fixed, it is obvious (over this example !) that the values of k_{12}, k_{21} and V_e also are uniquely defined. QED.
In practice, the function x_1 is known from a file of measures and one can try to numerically estimate the values of its first and its second derivative. If the measures are free of noise, the first derivative can be quite accurately estimated but this is usually not the case for the second derivative. To overcome these difficulties due to numerical approximations, one builds an overdetermined linear system that one solves by means of *linear* least squares. Over the example, one gets the following values:

$$[k_{12}+k_{21}]=2.1,\quad [V_e]=87.29,\quad [k_{21}\,V_e]=144.01.$$

The values of the blocks of parameters being known, one still has to recover the values of the parameters by solving the above algebraic system. Over this example, it is very easy and one gets:

$$V_e=87.29,\quad k_{12}=0.45,\quad k_{21}=1.65.$$

The above values can now be used as a starting point for the purely numerical method. Still over the example, one gets the correct parameters values:

$$V_e=101,\quad k_{12}=0.5,\quad k_{21}=3.$$

3.4 Issues and Implementation

In general, there is no guarantee that the first estimation provided by the symbolic-numeric method leads the purely numerical method in the global minimum. Estimating parameters only makes sense for models at least *locally identifiable*. However, testing this property does not raise any difficulty. Some seminumerical algorithms are available [64]. These are probabilistic algorithms for which the failure probability is known and can be decreased up to any value. Numerically estimating the derivatives raises

an important difficulty. To overcome it, a good method consists in converting the differential equations as integral equations as suggested in [22]. Under some conditions, integral equations are less sensitive to the noise than differential equations. There exists another important difficulty: there may exist algebraic relations between the parameters blocks. There is no such relation over the example. But assume, for the sake of the explanation, that the computed differential polynomial involves the three following blocks of parameters so that the third block is the product of the two first ones:

$$[V_e], \quad [k_{21}], \quad [V_e\, k_{21}].$$

There is no doubt that the numerical values produced during the resolution of the linear overdetermined system would not satisfy this relation. This would imply that the final algebraic system to solve in order to get the values of the parameters would be inconsistent. A way to overcome this problem consists in applying a nonlinear least squares method to solve the algebraic system. But then one needs to provide a first estimation of the parameters values: the problem to be overcome ! A symbolic method, based on algebraic elimination would be much more interesting. Indeed, it would provide the desired solution and could also compute the number of solutions of the algebraic system. It would solve in the same time the problem of estimating the parameters values and the problem of the identifiability of the model. Is it reasonable to try to apply algebraic elimination here ? One may think so, provided that many model variables are observed (at least one half). In this case, the differential elimination is fast and the parameters blocks are small: the algebraic elimination should be cheap.

A first draft of the above method was implemented in the *LEPISME* project [6]. The *Gnu Scientific Library* was used to perform the numerical methods. The *BLAD* libraries were used to perform the differential elimination. The method is difficult to implement in a satisfactory way: it involves many different steps. Each of these steps can be performed using a few different methods. When any method fails, it is difficult to provide synthetic informations on the failure to the user. In *BLAD*, instead of *Rosenfeld-Gröbner*, the more specialized and more efficient *PARDI* algorithm is used [11]. It takes advantage of the fact that the model equations generate a differential prime ideal and already form a characteristic set of this ideal w.r.t. some orderly ranking. It avoids all the discussions that *Rosenfeld-Gröbner* would perform and always computes only one characteristic set.

3.5 Prospects

In spite of all the difficulties, the project is being continued[5]: even in the case the symbolic-numeric method fails, the purely numerical method is still available. The existing method is thus improved. The use of the *BLAD* libraries is particularly interesting here for they permit to bound in advance the time and the memory allocated to the symbolic part of the symbolic-numeric method. Observe that the limitations are often due to the numerical part of the computations.

[5]The author is getting involved in a project which aims at applying this method for modelling the biosynthesis of fatty acids and oil in oilseed embryos.

4 Model Reduction

The green alga *ostreococcus tauri* (Figure 4.1) was discovered in 1994 in the *Étang de Thau*, in the south of France. It is the minimal non parasitic known organism. Its genom, constituted of 11 millions of pairs of bases was published in 2006. Though very simple, this unicellular organism is endowed by a *circadian clock*[1]. See [51] for an historical perspective essay on circadian clocks and [28, Chapter 9] or [32] for more general texts about oscillations in biology. This clock permits the alga to raise itself at the top of the water before the sunrise. The alga is one of the main objects of study of the the *Observatoire Océanologique de Banyuls*.

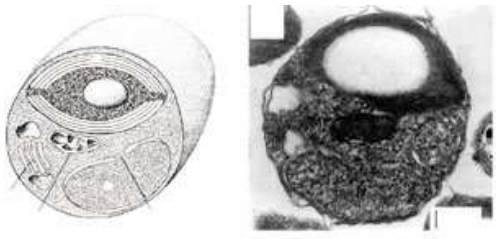

Figure 4.1: Ostreococcus tauri: a nucleus with a hole (bottom right), a chloroplast (top) with an amide ball (white spot), a Golgi apparatus (bottom left) and a mitochondry (center). The size is about one micrometer.

The author has been involved for two years in a pluridisciplinary working group (including computer scientists, physicists and biologists), led by François-Yves Bouget of the *Observatoire océanologique de Banyuls* for the biological part and Marc Lefranc of the *nonlinear dynamics* team for the physics and computer science part. This working group aims at modelling the cell division cycle of *ostreococcus tauri*. Our first goal has been to try to model the circadian clock of *ostreococcus tauri* which controls[2] the division cycle. In the genom of the green alga, two genes (named *TOC* and *CCA1*) were identified. They are known to be central components of clocks. We have thus been seeking a model under the form of a system of parametric ordinary differential equations, describing a two genes regulatory network and producing oscillating trajectories. We have very quickly met the following difficulty: many systems of parametric ordinary differential equations have integral curves which do not oscillate at all and, even the ones which have oscillating integral curves, only have such curves for very restricted ranges of parameters. Our problem can thus be reformulated as follows: *given a system of parametric ordinary differential equations, does there exist ranges of parameters w.r.t. which integral curves oscillate ?*

[1] A circadian clock is a clock the period of which is about 24 hours. The qualifier is built from *circa* (around) and *dies* (day).

[2] This is our simplifying working assumption. The clock itself might actually very well be regulated by the division cycle.

This problem can be addressed by looking for conditions on parameters which produce *Hopf bifurcations* [38, Chapter 11]. This approach was recently studied in the computer algebra community [27, 77, 35, 34]. It applies the Routh-Hurwitz criterion [36, Section I.13] and involves non differential elimination. It is not discussed in this paper. Another approach consists in applying the Poincaré-Bendixson theorem [38, Chapter 12] together with *differential elimination*. It was applied by members of the biology community in [72] over an abstract two genes regulatory network. This is the approach described in this chapter[3].

What does the Poincaré-Bendixson theorem state and how can it be applied in this context ? Roughly speaking, the theorem states that, if the integral curves of an autonomous ordinary differential system in *two dependent variables* stay in a bounded area and if this area does not involve any stable steady point then this area involves limit cycles. Limit cycles correspond to oscillating trajectories. Where is differential elimination involved ? The initial model (section 4.1) involves seven dependent variables. The idea consists in approximating it by a *reduced model* of two ordinary differential equations in two variables by means of *model reduction*. Differential elimination permits to simplify[4] the reduced model (section 4.2). The application of the Poincaré-Bendixson theorem is afterwards pretty straightforward. Indeed, in biology, trajectories of variables are always bounded. So are the ones of the reduced model, at least for parameters values which are biologically consistent (positivity is the least requirement). The steady points of the reduced model can be computed by algebraic elimination (e.g. Gröbner bases methods). There are three steady points but only one of them correspond to positive values of the variables (the other ones are discarded). Its stability can be studied by linearizing the model in the neighborhood of the steady point: the point is unstable if and only if at least one of the eigenvalues of the coefficients matrix J of the linearized system has a positive real part [36, Section I.13]. The conditions on parameters values which make the reduced system oscillate correspond thus to conditions on parameters values which make the trace and the determinant of the matrix J (which is 2×2) both positive. These parameters ranges make the reduced system oscillate. Do they make the initial model oscillate ? Yes ... provided that the model reduction is a good reduction ! This theoretically very difficult question can actually be checked, as in [72], by numerically integrating the initial model for many different parameters values picked in the estimated parameters ranges.

4.1 The Initial Model

This section describes the initial abstract model of [72]. The model involves two genes: an activator $\mathscr{A}$ and a repressor $\mathscr{R}$. These genes get transcribed into two mRNA M_A and M_R. The mRNAs then get translated into proteins A and R. Protein A can fix itself on the promotors of both genes $\mathscr{A}$ and $\mathscr{R}$, speeding up both transcription rates. The

[3] The author would like to thank Natacha Skrzypczak: an important part of the following analysis was initiated by her in [69].

[4] The author of [72] did not actually use any differential elimination method: they simplified their system interactively with MATHEMATICA. As shown later, the use of a differential elimination method permits to improve their result.

two proteins A and R can react together and form a complex C. Intuitively, one sees that the action of gene $\mathscr{A}$ consists in speeding up the reaction by producing protein A while the action of gene $\mathscr{R}$ consists in slowing down it by producing protein R which catches A to form the complex.

The seven model variables. The variables M_A and M_R denote the concentrations of mRNA transcribed from genes $\mathscr{A}$ and $\mathscr{R}$. The variables A, R and C denote the concentrations of the corresponding proteins. For each gene, one needs to introduce a variable to distinguish the case where protein A is bound to its promotor from the case where protein A is not bound to its promotor[5]. This variable is not a concentration. It should rather be considered as a probability or a mean value: the variable D_A corresponds to the gene $\mathscr{A}$. The value 1 indicates that protein A is bound to the promotor of $\mathscr{A}$. The value 0 indicates that protein A is not bound to the promotor. A similar variable D_R is introduced for gene $\mathscr{R}$. There are 15 parameters, denoted by Greek letters.

The model equations. They are derived from a picture. Since the complete picture might be a bit difficult to interpret for casual readers, it is explained and built piece by piece. Picture 4.2 describes the possible binding of protein A on the promotors of genes $\mathscr{A}$ and $\mathscr{R}$.

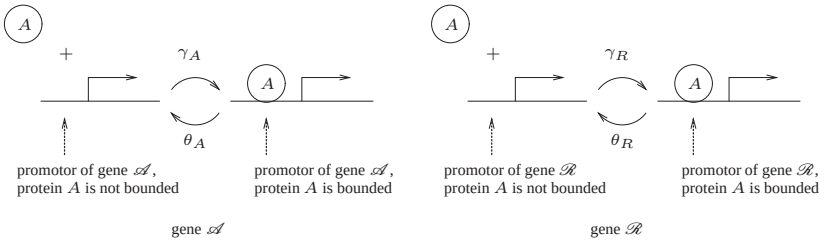

Figure 4.2 The two possible states of genes $\mathscr{A}$ and $\mathscr{R}$.

The corresponding model equations[6] are given below. The "plus signs" in the diagram indicate that the binding rate of protein A is proportional to the product of the concentration of A by the variables D_A and D_R. It is a variant of the *mass action law*, variables D_A and D_R being handled as concentrations[7]. Observe that one temporarily

[5]The *promotor* of a gene is an area located in front of the gene. For a gene to be transcribed into mRNA, it is necessary that some protein binds itself to the gene promotor. Many different proteins may play this role. In this model, it is implicitly assumed that some unspecified proteins different from A may bind themselves to the promotors of the two genes, but with more difficulty than A so that the transcription rates of the two genes are higher when A is bound than when A is not bound.

[6]The model given in [72] involves nine variables instead of seven: two extra variables were introduced to avoid the $(1 - D)$ terms.

[7]One may wonder why differential equations are used to model such phenomenons while stochastic equations might better correspond to the reality. An answer is that the qualitative analysis of the model is much easier with the rich theory of systems of ordinary differential equations than with stochastic equations. Of course, the conclusions derived from the differential model should be validated afterwards by stochastic simulations as the authors of [72] actually do.

omits the differential equation which describes the evolution of A because it would be incomplete at this step.

$$\dot{D}_A = -\gamma_A\, A\, D_A + \theta_A\,(1 - D_A), \quad \dot{D}_R = -\gamma_R\, A\, D_R + \theta_R\,(1 - D_R).$$

The leftmost part of Figure 4.3 shows that gene $\mathscr{A}$ gets transcribed into mRNA M_A at different rates depending on whether protein A is bound or not to its promotor. The mRNA M_A can be degraded at rate δ_{M_A}. The rightmost part of the figure shows a symmetric phenomenon for gene $\mathscr{R}$. The corresponding model equations are given

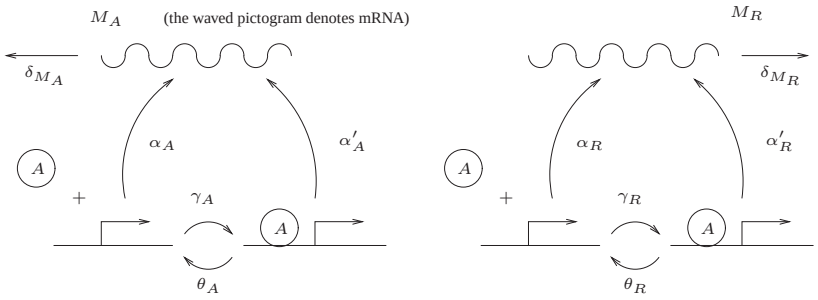

Figure 4.3 Transcriptions of the two genes into mRNA.

below. They enlarge the above set of two differential equations. Observe that the terms $\alpha_A\, D_A + \alpha'_A\,(1 - D_A)$ and $\alpha_R\, D_R + \alpha'_R\,(1 - D_R)$ are not subtracted to the right-hand sides of the differential equations which describe the evolutions of D_A and D_R (contrarily to what is usually done when translating chemical reactions into differential equations) since genes are not consumed by transcriptions.

$$\dot{M}_A = \alpha'_A\,(1 - D_A) + \alpha_A\, D_A - \delta_{M_A}\, M_A, \quad \dot{M}_R = \alpha'_R\,(1 - D_R) + \alpha_R\, D_R - \delta_{M_R}\, M_R.$$

The complete diagram is given in Figure 4.4. It indicates that mRNAs M_A and M_B get translated into proteins A and R. Since translations do not consume mRNA, the terms $\beta_A\, M_A$ and $\beta_R\, M_R$ are not subtracted to the right-hand sides of the two differential equations above. Figure 4.4 also shows that proteins A and R can react together to form[8] a complex C. The complex C may break, producing back protein R. There are degradations rates for proteins A and R. The new model equations are given below. They enlarge the set of four equations previously built.

$$\dot{C} = \gamma_C\, A\, R - \delta_A\, C, \quad \dot{R} = \beta_R\, M_R - \gamma_C\, A\, R + \delta_A\, C - \delta_R\, R,$$
$$\dot{A} = \theta_A\,(1 - D_A) + \theta_R\,(1 - D_R) + \beta_A\, M_A - (\gamma_A\, D_A + \gamma_R\, D_R + \gamma_C\, R + \delta_A)\, A.$$

4.2 Reduction of the Model

One tries to approximate the model built in section 4.1, which involves seven parametric ordinary differential equations, in seven variables, by a system of two parametric

[8]This *dimerization* of the two proteins does not seem to occur in the context of *ostreococcus tauri*. This causes a difficulty to apply the model of [72] to the green alga for the oscillating behaviour of the model seems to be strongly related to the dimerization.

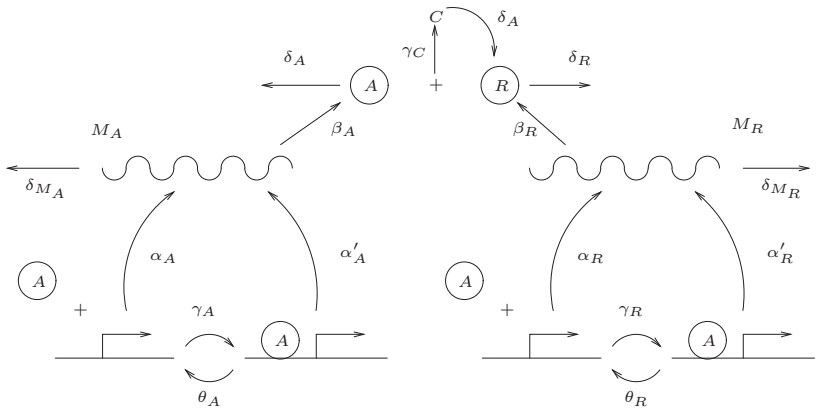

Figure 4.4 The complete diagram.

ordinary differential equations, in two variables. To eliminate five variables, the idea consists in separating the seven variables into a set of two *slow* variables, a set of five *fast* variables and to proceed to a *steady state approximation* [36, Section I.16]. Roughly speaking, here is the idea: consider a differential system of the following form, where ε denotes a *small* positive constant:

$$\dot{x} = f(x,\, y), \quad \varepsilon\, \dot{y} = g(x,\, y).$$

Over a generic point $(x,\, y) \in \mathbb{R}^2$ and, in particular, in the neighborhood of the initial conditions, the speed of y is high and thus rapidly approaches an area where $g(x,\, y) \simeq 0$. It is thus reasonable to approximate such a system by the following one:

$$\dot{x} = f(x,\, y), \quad 0 = g(x,\, y)$$

which mixes differential and algebraic equations. The study of such systems is not easy, in particular when there are many different algebraic equations $g_i = 0$. Numerical integrators cannot usually guarantee that the computed integral curves stay on the algebraic variety defined by the algebraic equations. For such systems, there may also exist hidden algebraic equations, consequences of the $g_i = 0$, which must be satisfied. Differential elimination is a tool which may simplify such systems and uncover these hidden equations. The authors of [72] decided that D_A, D_R, M_A, M_R and A are fast and R and C are slow. They were thus led to study the following differential algebraic

system:

$$\begin{aligned}
\dot{D}_A &= \theta_A(1-D_A) - \gamma_A\, D_A\, A,\\
\dot{D}_R &= \theta_R\,(1-D_R) - \gamma_R\, D_R\, A,\\
\dot{M}_A &= \alpha'_A\,(1-D_A) + \alpha_A\, D_A - \delta_{M_A}\, M_A,\\
\dot{M}_R &= \alpha'_R\,(1-D_R) + \alpha_R\, D_R - \delta_{M_R}\, M_R,\\
\dot{A} &= \theta_A\,(1-D_A) + \theta_R\,(1-D_R) + \beta_A\, M_A\\
&\qquad -(\gamma_A\, D_A + \gamma_R\, D_R + \gamma_C\, R + \delta_A)\, A,\\
\dot{R} &= \beta_R\, M_R - \gamma_C\, A\, R + \delta_A\, C - \delta_R\, R,\\
\dot{C} &= \gamma_C\, A\, R - \delta_A\, C.
\end{aligned}$$

The authors of [72] did actually perform a differential elimination process over the above example, without stating the words *differential elimination*. They did it interactively, using MATHEMATICA. The *diffalg* package of MAPLE can indeed perform the same task. We are somehow looking for a differential system involving only R and C. A natural ranking to choose is the following one, which eliminates the fast variables:

$$(\textit{fast variables}) \gg (\textit{slow variables}).$$

However, to avoid a pointless expression swell and to obtain exactly the same result as [72], it is better to keep the fast variable A in the set of the slow variables. Here are the coresponding MAPLE commands:

```
syst := [thetaA*(1 - DA) - gammaA*DA*A,
         thetaR*(1 - DR) - gammaR*DR*A,
         alphaAp*(1 - DA) + alphaA*DA - deltaMA*MA,
         alphaRp*(1 - DR) + alphaR*DR - deltaMR*MR,
         thetaA*(1 - DA) + thetaR*(1 - DR) + betaA*MA
            - (gammaA*DA + gammaR*DR + gammaC*R + deltaA)*A,
         R[t] - (betaR*MR - gammaC*A*R + deltaA*C - deltaR*R),
         C[t] - (gammaC*A*R - deltaA*C)]:

K := field_extension (transcendental_elements = [thetaA, thetaR,
    gammaA, gammaR, gammaC, alphaA, alphaAp, alphaR, alphaRp,
    betaA, betaR, deltaA, deltaR, deltaMA, deltaMR]):
Ring := differential_ring (derivations = [t],
         field_of_constants = K,
         ranking = [[DA, DR, MA, MR], [A, R, C]]);

ideal := Rosenfeld_Groebner (syst, Ring):
```

The list *ideal* only involves one characteristic set, involving seven equations. The three last equations only depend on R, C, A, $\dot{R}$ and $\dot{C}$. They have the following form:

$$\begin{aligned}
\dot{R} &= \textit{a rational fraction},\\
\dot{C} &= \gamma_C\, A\, R - \delta_A\, C,\\
0 &= (\gamma_A\, \delta_{M_A}\, \delta_A + \gamma_A\, \delta_{M_A}\, \gamma_C\, R)\, A^2 +\\
&\qquad (\delta_A\, \theta_A\, \delta_{M_A} + \theta_A\, \delta_{M_A}\, \gamma_C\, R - \alpha'_A\, \gamma_A\, \beta_A)\, A - \beta_A\, \theta_A\, \alpha_A.
\end{aligned}$$

Observe that the above system is not so easy to integrate numerically: solving the third equation implies to choose a root of a degree two polynomial. Here, the choice is straightforward for A, being a concentration, needs to be positive and the equation always has only one positive root[9]. However, this argument needs some understanding of the system: the user has to manipulate the equation and solve it explicitly in order to select the positive root. Having a differential elimination algorithm at hand permits us however to try to compute many different representations of the same system. In particular, if one tries the following ranking, obtained by permuting A, R and C in the second block, one gets a simpler representation[10]:

```
Ring := differential_ring (derivations = [t],
         field_of_constants = K,
         ranking = [[DA, DR, MA, MR], [R, C, A]]):
ideal := Rosenfeld_Groebner (syst, Ring);
```

The list *ideal* only involves one characteristic set. The last three equations provide another presentation of the reduced model with two ordinary differential equations and a degree one algebraic equation:

$$\dot{C} = \textit{a rational fraction},\ \dot{A} = \textit{a rational fraction},\ R = \textit{a rational fraction}.$$

Moreover, the variable R does not appear anywhere in the two differential equations since any occurence of R would have been replaced by the right-hand side of the last equation. One can thus just omit the third, algebraic, equation.
The above system might be surprising for readers not familiar with steady state approximations. Indeed, the reduced model was obtained by letting the speeds of the fast variables (including A) equal to zero. How is it then possible to end up with a differential equation defining a nonzero speed for A ? The answer comes from the fact that the above sentence is wrong: the speed of A was not set[11] to zero ! Indeed, the differential equations describing the evolutions of the fast variables were removed. The resulting system of two ordinary differential equations was just specialized on the algebraic variety defined by the right-hand sides of the removed equations.

4.3 Prospects

Ranges of parameters values which make the reduced and the initial model oscillate are given in [72] but the authors do not describe the method they applied to compute these ranges. Clearly, one now needs a method able to automatically derive ranges of

[9] The first coefficient is positive and the last coefficient is negative: the number of positive real roots is at least one. Now, whatever the sign of the central coefficient, the number of sign changes is one. By Descartes rule of sign [3], the number of positive real roots is at most one. The polynomial thus always has exactly one positive real root.

[10] Observe that there are other possible permutations over the second block of variables. Most of them lead to untractable computations. This example illustrates the need of software able to try many different reasonable rankings with a time limit. The *BLAD* libraries are designed to offer such a functionnality.

[11] The interested reader may try to apply *Rosenfeld-Gröbner* over the initial system enlarged with the five ordinary differential equations setting to zero the speeds of the five fast variables. One gets an inconsistent system.

parameters from the sign conditions on the trace and the determinant of the computed matrix. Observe that even heuristic methods would be helpful and that the derived ranges of parameters do not need to be complete in any sense. Though such methods can certainly be designed in theory (e.g. based on interval arithmetic [39]), the choice is not so easy in practice. It must still be done in order to get an automatic method which would help modelling the circadian clock of *ostreococcus tauri*. Last, observe that it would be very interesting to compare the approach based on the Poincaré-Bendixson theorem, and the one based on the direct application of the Routh-Hurwitz criterion over the initial model. This study also is still in progress.

5 Conclusion

Differential elimination is a tool which may play a real role to improve some applied mathematics methods. As illustrated in section 2.3 and 4, it permits to reduce the differentiation index of differential-algebraic systems. It permits also to compute different representations of the same system. Both features may help designing better numerical integrators. Differential elimination may help guessing good starting points for Newton methods (section 3). It may also be involved in the qualitative analysis of dynamical system for it permits to simplify these systems after model reduction (section 4). These examples show that differential elimination is complementary to numerical methods. It is interesting here to compare the non differential and the differential elimination. From an algorithmic point of view, both theories are very close to each other. From the applications standpoint, the situations are very different. In the non differential setting, one may hope to bypass all numerical methods. For instance, in the zerodimensional case, there exists symbolic algorithms [1, 63] able to isolate the real roots of large polynomials. It thus makes sense to compute large Gröbner bases or characteristic sets. In the differential setting however, no such algorithms are known. Cooperating with numerical methods is thus mandatory. Now, differential systems which are considered as difficult from the numerical point of view are actually very small and very easy from the symbolic one (see [37, Section IV.1]). It thus may not really make sense to compute large differential characteristic sets. Note that this observation is an argument which minimizes the importance of the terrible worst case complexity of differential methods ! The examples considered in this paper also show that differential elimination only play very local roles in the different processes: it helps but may quite often be bypassed. Since moreover, the theory is rather difficult and usually not taught in traditional university courses, it seems very important to develop easy to use software components. The *BLAD* libraries are an attempt in that direction.

Bibliography

[1] Alkiviadis G. Akritas, *There is no Uspensky's method*, proceedings of ISSAC'86, 1986.

[2] Stefania Audoly, Giuseppina Bellu, Leontina D'Angio, Maria Pia Saccomani, and Claudio Cobelli, *Global Identifiability of Nonlinear Models of Biological Systems*, IEEE Transactions on Biomedical Engineering 48 (2001), pp. 55–65.

[3] Saugata Basu, Richard Pollack, and Marie-Françoise Roy, *Algorithms in Real Algebraic Geometry*, Algorithms and Computation in Mathematics, vol. 10, Springer Verlag, 2003.

[4] François Boulier, *The BLAD libraries*, `http://www.lifl.fr/~boulier/BLAD`, 2004.

[5] ———, *Réécriture algébrique dans les systèmes d'équations différentielles polynomiales en vue d'applications dans les Sciences du Vivant*, May 2006, Mémoire d'habilitation à diriger des recherches, Université Lille I, LIFL, 59655 Villeneuve d'Ascq, France.

[6] François Boulier, Lilianne Denis-Vidal, Thibaut Henin, and François Lemaire, *LÉPISME*, presented at the ICPSS conference, 2004, submitted to the Journal of Symbolic Computation.

[7] François Boulier, *Étude et implantation de quelques algorithmes en algèbre différentielle*, Ph.D. thesis, Université Lille I, 59655, Villeneuve d'Ascq, France, 1994.

[8] ———, *Efficient computation of regular differential systems by change of rankings using Kähler differentials*, Tech. report, Université Lille I, 59655, Villeneuve d'Ascq, France, November 1999, (ref. LIFL 1999–14, presented at the MEGA2000 conference).

[9] François Boulier, Daniel Lazard, François Ollivier, and Michel Petitot, *Representation for the radical of a finitely generated differential ideal*, Proceedings of ISSAC'95 (Montréal, Canada), 1995, pp. 158–166.

[10] ———, *Computing representations for radicals of finitely generated differential ideals*, Tech. report, Université Lille I, LIFL, 59655, Villeneuve d'Ascq, France, 1997, (ref. IT306, december 1998 version published in the habilitation thesis of Michel Petitot).

[11] François Boulier, François Lemaire, and Marc Moreno Maza, *PARDI !*, proceedings of ISSAC'01 (London, Ontario, Canada), 2001, pp. 38–47.

[12] François Boulier and François Lemaire, *Computing canonical representatives of regular differential ideals*, proceedings of ISSAC 2000 (St Andrews, Scotland), 2000, pp. 37–46.

[13] François Boulier, François Lemaire, and Marc Moreno Maza, *Well known theorems on triangular systems and the D^5 principle*, Proceedings of Transgressive Computing 2006 (Granada, Spain), 2006, pp. 79–91.

[14] Driss Bouziane, Abdelillah Kandri Rody, and Hamid Maârouf, *Unmixed–Dimensional Decomposition of a Finitely Generated Perfect Differential Ideal*, Journal of Symbolic Computation 31 (2001), pp. 631–649.

[15] Manuel Bronstein, *Integration and Differential Equations in Computer Algebra*, Programmirovanie 5 (1992), pp. 26–44.

[16] Bruno Buchberger, *Ein Algorithmus zum Auffinden der Basiselemente des Restklassenringes nach einem nulldimensionalen Polynomideals*, Ph.D. thesis, University of Innsbruck, Austria, Math. Institute, Austria, 1966, English translation in [17].

[17] ———, *An Algorithm for Finding the Basis Elements in the Residue Class Ring Modulo a Zero Dimensional Polynomial Ideal*, Journal of Symbolic Computation (Special Issue on Logic, Mathematics, and Computer Science: Interactions) 41 (2006), pp. 475–511.

[18] Alexandru Buium and Phyllis Cassidy, *Differential Algebraic Geometry and Differential Algebraic Groups: From Algebraic Differential Equations To Diophantine Geometry*, pp. 567–636, Amer. Math. Soc., Providence, RI, 1998.

[19] Giuseppa Carra-Ferro, *Gröbner bases and differential ideals*, Notes of AAECC 5 (Menorca, Spain), Springer Verlag, 1987, pp. 129–140.

[20] S. Demignot and D. Domurado, *Effect of prosthetic sugar groups on the pharmacokinetics of glucose-oxidase*, Drug Design Deliv. 1 (1987), pp. 333–348.

[21] Jan Denef and Leonard Lipshitz, *Power Series Solutions of Algebraic Differential Equations*, Mathematische Annalen 267 (1984), pp. 213–238.

[22] Lilianne Denis-Vidal, *Identifiabilité de modèles non linéaires paramétriques de dynamiques classiques et à retard. Planification d'expériences et estimation de paramètres*, Mémoire d'Habilitation à Diriger des Recherches, Université de Technologie de Compiègne, may 2004.

[23] Lilianne Denis-Vidal, Ghislaine Joly-Blanchard, and Céline Noiret, *System identifiability (symbolic computation) and parameter estimation (numerical computation)*, Numerical Algorithms, vol. 34, 2003, pp. 282–292.

[24] Sette Diop, *Elimination in Control Theory*, Mathematics of Control, Signal and Systems 4 (1991), pp. 17–42.

[25] ———, *Differential algebraic decision methods and some application to system theory*, Theoretical Computer Science 98 (1992), pp. 137–161.

[26] Sette Diop and Michel Fliess, *Nonlinear observability, identifiability, and persistent trajectories*, Proc. 30th CDC (Brighton), 1991, pp. 714–719.

[27] M'Hammed El Kahoui and Andreas Weber, *Deciding Hopf bifurcations by quantifier elimination in a software–component architecture*, Journal of Symbolic Computation 30 (2000), pp. 161–179.

[28] C. P. Fall, E. S. Marland, J. M. Wagner, and John J. Tyson, *Computational Cell Biology*, Interdisciplinary Applied Mathematics, vol. 20, Springer Verlag, 2002.

[29] Jean-Charles Faugère, Patricia Gianni, Daniel Lazard, and Teo Mora, *Efficient computation of Gröbner bases by change of orderings*, Journal of Symbolic Computation 16 (1993), pp. 329–344.

[30] Michel Fliess, *Automatique et corps différentiels*, Forum Math. 1 (1989), pp. 227–238.

[31] Michel Fliess, Jean Lévine, Philippe Martin, and Pierre Rouchon, *Index and Decomposition of Nonlinear Implicit Differential Equations*, Proceedings of IFAC, 1995, pp. 43–48.

[32] Jean-Pierre Françoise, *Oscillations en biologie*, Mathématiques et Applications, vol. 46, Springer Verlag, 2005.

[33] Giovanni Gallo, Bubaneshwar Mishra, and François Ollivier, *Some constructions in rings of differential polynomials*, Lecture Notes in Computer Science, vol. 539, pp. 171–182, , Montréal, Canada, 1991.

[34] Karin Gatermann, Markus Eiswirth, and Anke Sensse, *Toric Ideals and graph theory to analyze Hopf bifurcations in mass action systems*, Journal of Symbolic Computation 40 (2005), pp. 1361–1382.

[35] Karin Gatermann and Serkan Hosten, *Computational algebra for bifurcation theory*, Journal of Symbolic Computation 40 (2005), pp. 1180–1207.

[36] Ernst Hairer, Syvert Paul Norsett, and Gerhard Wanner, *Solving ordinary differential equations I. Nonstiff problems*, 2. ed., Springer Series in Computational Mathematics, vol. 8, Springer–Verlag, New York, 1993.

[37] Ernst Hairer and Gerhard Wanner, *Solving ordinary differential equations II. Stiff and Differential–Algebraic Problems*, 2. ed., Springer Series in Computational Mathematics, vol. 14, Springer–Verlag, New York, 1996.

[38] Jack K. Hale and Hüseyin Koçak, *Dynamics and Bifurcations*, Texts in Applied Mathematics, vol. 3, Springer–Verlag, New York, 1991.

[39] Eldon Hansen, *Global Optimization Using Interval Analysis*, Marcel Dekker Inc., New York, 1992.

[40] Évelyne Hubert, *Factorization free decomposition algorithms in differential algebra*, Journal of Symbolic Computation 29 (2000), pp. 641–662.

[41] ———, *Notes on triangular sets and triangulation–decomposition algorithm II: Differential Systems*, Symbolic and Numerical Scientific Computing 2001 (2003), pp. 40–87.

[42] Mickael Kalkbrener, *A Generalized Euclidean Algorithm for Computing Triangular Representations of Algebraic Varieties*, Journal of Symbolic Computation 15 (1993), pp. 143–167.

[43] Ellis Robert Kolchin, *Differential Algebra and Algebraic Groups*, Academic Press, New York, 1973.

[44] Klaus Kühnle and Ernst Wilhelm Mayr, *Exponential Space Computation of Gröbner Bases*, proceedings of ISSAC'96 (Zürich, Switzerland), 1996, pp. 63–71.

[45] Daniel Lazard, *A new method for solving algebraic systems of positive dimension*, Discrete Applied Mathematics 33 (1991), pp. 147–160.

[46] ———, *Solving Zero–dimensional Algebraic Systems*, Journal of Symbolic Computation 13 (1992), pp. 117–131.

[47] François Lemaire, *An orderly linear PDE system with analytic initial conditions with a non analytic solution*, Special Issue on Computer Algebra and Computer Analysis, Journal of Symbolic Computation 35 (2003), pp. 487–498.

[48] Ziming Li and Dongming Wang, *Coherent, regular and simple systems in zero decompositions of partial differential systems*, Systems Science and Mathematical Sciences 12 (1999), pp. 43–60.

[49] L. Ljung and S. T. Glad, *On global identifiability for arbitrary model parametrisations*, Automatica 30 (1994), pp. 265–276.

[50] Elizabeth L. Mansfield, *Differential Gröbner Bases*, Ph.D. thesis, University of Sydney, Australia, 1991.

[51] C. Robertson McClung, *Plant Circadian Rhythms*, The Plant Cell 18 (2006), pp. 792–803.

[52] Marc Moreno Maza, *Calculs de Pgcd au–dessus des Tours d'Extensions Simples et Résolution des Systèmes d'Équations Algébriques*, Ph.D. thesis, Université Paris VI, France, 1997.

[53] ———, *On Triangular Decompositions of Algebraic Varieties*, Tech. report, NAG, 2000, (presented at the MEGA2000 conference, submitted to the JSC).

[54] Sally Morrison, *Yet another proof of Lazard's lemma*, private communication, december 1995.

[55] ———, *The Differential Ideal* $[P] : M^{\infty}$, Journal of Symbolic Computation 28 (1999), pp. 631–656.

[56] Céline Noiret, *Utilisation du calcul formel pour l'identifiabilité de modèles paramétriques et nouveaux algorithmes en estimation de paramètres*, Ph.D. thesis, Université de Technologie de Compiègne, 2000.

[57] François Ollivier, *Le problème de l'identifiabilité structurelle globale : approche théorique, méthodes effectives et bornes de complexité*, Ph.D. thesis, École Polytechnique, Palaiseau, France, 1990.

[58] Gregory J. Reid, Ping Lin, and Allan D. Wittkopf, *Differential Elimination–Completion Algorithms for DAE and PDAE*, Studies in Applied Mathematics 106 (2001), pp. 1–45.

[59] Gregory J. Reid, Allan D. Wittkopf, and Alan Boulton, *Reduction of systems of nonlinear partial differential equations to simplified involutive forms*, European Journal of Applied Math. (1996), pp. 604–635.

[60] Charles Riquier, *Les systèmes d'équations aux dérivées partielles*, Gauthier–Villars, Paris, 1910.

[61] Joseph Fels Ritt, *Differential Algebra*, Dover Publications Inc., New York, 1950, Available at `http://www.ams.org/online_bks/coll33`.

[62] Azriel Rosenfeld, *Specializations in differential algebra*, Trans. Amer. Math. Soc. 90 (1959), pp. 394–407.

[63] Fabrice Rouillier and Paul Zimmermann, *Efficient isolation of a polynomial real roots*, Journal of Computational and Applied Mathematics 162 (2004), pp. 33–50.

[64] Alexandre Sedoglavic, *A Probabilistic Algorithm to Test Local Algebraic Observability in Polynomial Time*, Journal of Symb. Comp. 33 (2002), pp. 735–755.

[65] Abraham Seidenberg, *Some basic theorems in differential algebra (characteristic p arbitrary)*, Trans. Amer. Math. Soc. 73 (1952), pp. 174–190.

[66] ———, *An elimination theory for differential algebra*, Univ. California Publ. Math. (New Series) 3 (1956), pp. 31–65.

[67] ———, *Abstract differential algebra and the analytic case*, Proc. Amer. Math. Soc. 9 (1958), pp. 159–164.

[68] Werner Markus Seiler, *Computer Algebra and Differential Equations. An Overview*, mathPAD 7 (1997), pp. 34–49, available at `http://www.iwr.uni-heidelberg.de/groups/compalg/seiler`.

[69] Natacha Skrzypczak, *Modélisation des réseaux de gènes : formalisme SBML2. Réduction des modèles biologiques*, Tech. report, Mémoire de Master 2. Université Lille I, France, 2005.

[70] Wu Wen Tsün, *On the foundation of algebraic differential geometry*, Mechanization of Mathematics, research preprints 3 (1989), pp. 2–27.

[71] Nathalie Verdière, *Identifiabilité de systèmes d'équations aux dérivées partielles semi–discrétisées et applications à l'identifiabilité paramétrique de modèles en pharmacocinétique ou en pollution*, Ph.D. thesis, Université de Technologie de Compiègne, France, 2005.

[72] José M. G. Vilar, Hao Yuan Kueh, Naama Barkai, and Stanislas Leibler, *Mechanisms of noise-resistance in genetic oscillators*, Proceedings of the National Academy of Science of the USA 99 (2002), pp. 5988–5992.

[73] Joachim von zur Gathen and Jürgen Gerhard, *Modern Computer Algebra*, Cambridge University Press, United Kingdom, 1999.

[74] Éric Walter, *Identifiability of State Space Models*, Lecture Notes in Biomathematics, vol. 46, Springer Verlag, 1982.

[75] Dongming Wang, *An elimination method for differential polynomial systems I*, Tech. report, LIFIA–IMAG, Grenoble, France, 1994.

[76] ———, *Elimination Practice: Software Tools and Applications*, Imperial College Press, London, 2003.

[77] Dongming Wang and Bican Xia, *Stability Analysis of Biological Systems with Real Solution Classification*, proceedings of ISSAC 05 (Beijing, China), 2005, pp. 354–361.

[78] Oscar Zariski and Pierre Samuel, *Commutative Algebra*, Van Nostrand, New York, 1958.

Index

Author information

François Boulier, LIFL, University Lille I, 59655 Villeneuve d'Ascq, France.
Email: boulier@lifl.fr